6 Password Weaknesses in Physical Security: Silliness in Three Acts

by Evan Sultanik

Dramatis Personæ

Disembodied Voice of Pastor Manul Laphroaig ..... Bard

Alice Feynman ..... Disciple of the Church of Weird Machines

Bob Schrute ..... Assistant to the Facility Security Officer

Havva al-Kindi ..... Alice's Old and Wise Officemate

The Ghost of Paul Erdős ..... Keeper of *The Book*

Act I: Memorize, Don't Compromise

Pastor: In the windowless bowels of a nondescript, Class A office building entrenched inside the Washington, D.C. beltway, we meet our heroine, Alice Feynman, lost on her way to a meeting with the Facility Security Officer.

Alice: Excuse me, which way is it to the security office?

Bob: You must be the new hire. Bob Schrute, assistant FSO. I can take you there right after I finish with this...

Alice: Alice. Nice to meet you. What're you doing?

Bob: Kaba Mas X-09 high security spin-lock. It's DSS-approved for use in our SCIFs. I'm resetting this one's passcode.

Alice: [*Blank Stare*]

Bob: U.S. Department of Defense (DoD) Defense Security Service (DSS). Sensitive Compartmented Information Facilities (SCIFs). The rooms where we are allowed to store and process classified information?

Alice: I see. I noticed those things all over this building.

Bob: They're ubiquitous. You'll see them anywhere in the country there's classified work going on. One on each door, and another on each safe. Super secure, too. Security in this office is no joke.

Alice: How do they work?

Bob: [*Throwing Alice the lock's manual.*] They run off of the electricity generated from spinning them, so you need to spin them a bit to get started. You see? The LCD on top shows you the current number. You enter three two-digit numbers. First one clockwise, second counter-clockwise, third clockwise, and then a final spin counter-clockwise to open. That's the passcode.

Alice: [*Flipping through the manual.*] Does each lock get a different passcode?

Bob: Yes. That's why we have this [*handing Alice a magnet stuck to the side of the door*].

Alice: Ah I see. It's a phone keypad. So you use a mnemonic to remember each passcode?

Bob: Exactly. [*Pointing to a poster on the wall with his own mugshot and memetic letters emblazoning "MEMORIZE, DON'T COMPROMISE", he sternly repeats that slogan:*] **Memorize, don't compromise**.

Alice: [*"Is this guy serious?" face.*]

Bob: You think you could crack it? FALSE. [*Flamboyantly produces a pocket calculator that had been hidden somewhere on his person.*] Three two-digit numbers. That's 100 times 100 times 100, so ... there are a million possible codes. I've set this to have a timeout of four minutes after each failed attempt. So, trying all possible combinations

would take . . . [*furiously punching at the calculator*] . . . almost eight years! We change each code once every couple months, so even if you could continuously try codes for eight hours a day, you'd have . . . [*more furious punching*] . . . about seven tenths of one percent chance of getting the code right.

Alice: [*Handing the manual back.*] I didn't see anything in here about an automatic lockout after too many failed attempts.

Bob: [*Pointing to his minuscule biceps.*] These provide the lockout.

Alice: Are you ready to take me to the security office now?

Bob: Fine.

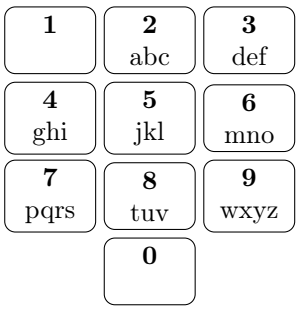
Act II: Surely You're Joking

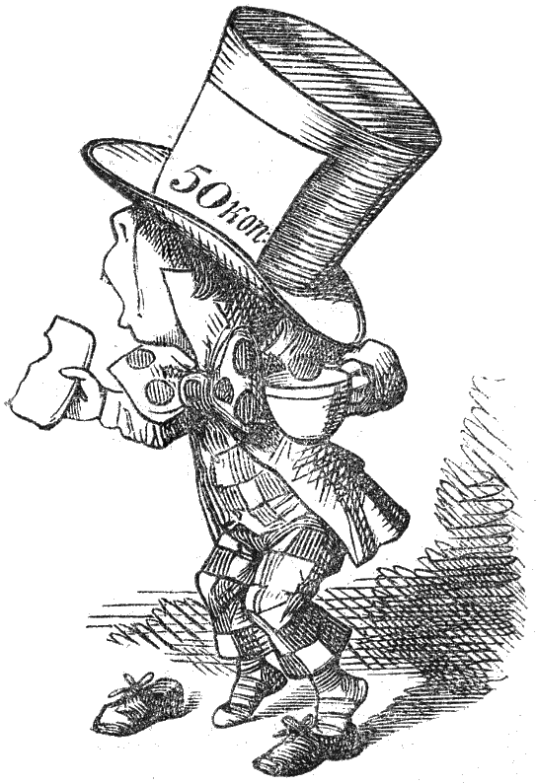
Pastor: Two weeks later, Alice has settled into her office, which she shares with Havva al-Kindi. She hasn't had a chance to play with those nifty locks at all yet; her clearance is still being processed. Most of her time is spent idling or doing busy-work while she waits to be approved to work on a real project.

Alice: [*On her desk phone*] Yes. Yes, no problem. By close of business today. No problem. Bye.

Pastor: As Alice hangs up the phone, she notices something odd about the keypad, and immediately remembers the magnet Bob had showed her.

Alice: [*Gets up and starts drawing on her whiteboard.*]





Havva: What are you doing?

Alice: Did you ever notice that the numbers zero and one don't have any letters on the phone?

Havva: Sure! You're probably too young to have ever used a rotary phone, right? Back when phone numbers were only seven digits long, the first two numbers represented the exchange, and a mnemonic was given to each exchange. [*Singing and tapping on her desk*] *Bum-dah-bum bah-duh-bum bahhh dummm! PEnnsylvania Six Five Thousand!* No? It was a big Glenn Miller hit! My parents used to play it all the time when I was a kid. That song is referring to the phone number for the Hotel Pennsylvania in New York, which to this day is still (212) PE6-5000.

Alice: Oh yeah! I went there once for HOPE.

Havva: Hope? Anyhow, for various reasons, the numbers zero and one were never used in exchanges, which meant they never occurred at the beginning of phone numbers, which meant they couldn't have letters associated with them.

ALICE: Interesting! [*Continuing on the whiteboard*] $8^6 = \ldots$ [*a pause to consult her computer*] 262144. $1 - 262144 \div 1000000 = \ldots$ 0.738. Wow! So, if there are only eight buttons with letters, that reduces the number of possible phone numbers associated with six-letter mnemonics by 74% compared to if all the buttons had letters!

HAVVA: I guess that's true. There are also certain phone numbers you'll never be able to have English mnemonics for, because the buttons for 5, 7, and 9 don't have any vowels. So you can't make a mnemonic for a phone number that only uses those three numbers.

ALICE: Wow, yeah, that's another $3^6 = \ldots$ [*quickly doing some math in her head this time*] 729 codes that don't have mnemonics.

HAVVA: Codes?

ALICE: Er, I mean "phone numbers."

HAVVA: I'll bet there are certain "codes" that don't have any English words associated with them. Plus, letters in English words don't all occur at the same frequency: It's much more likely that a word will have the letter "e" than it will have the letter "x."

ALICE: [*Opens up a terminal on her computer.*]

```
$ grep '^.\{6\}$' /usr/share/dict/words | wc -l
    17706
$ echo `!!` / 1000000 | bc -l
.01770600000000000000
```

PASTOR: And thus, Alice had discovered that fewer than 2% of the million possible codes actually map to English words.

ALICE: [*Once again at the whiteboard.*]

HA	CK	ER
42	25	34

[*Back at the computer.*]

```
$ grep -i '^.\{4\}er$' /usr/share/dict/words \
  | wc -l
    1562
```

About 10% of six-letter English words end with the letters "ER"!

[*Back at the board, with long pauses.*]

DO	SA	GE
36	72	43
EN	RA	GE
36	72	43
FO	RA	GE
36	72	43
FO	RB	ID
36	72	43

PASTOR: And many words share the same code. In fact, Alice quickly wrote a script to count the number of unique codes possible from six-letter English words[17].

ALICE: There are only 14684 possible codes to check! That would take ... only about 40 days to brute-force crack!

Act III: The Book

PASTOR: Later that day, Alice is at her favorite dive, decompressing with some of her side projects.

PAUL: [*Sits down next to Alice at the bar. Wheel of Fortune is playing on an ancient CRT.*] Television is something the Russians invented to destroy American education.

ALICE: [*Tippling a brown liquor, neat, while working on her laptop. Paul's comment draws her attention to the TV. Alice notices that some letters are given away "for free" and remembers what Havva had said about letter frequency. She quickly grabs her notebook and jots down the letters as a reminder.*] R, S, T, L, N, E.

PAUL: [*Noticing Alice's notebook.*] Yes, these are very common letters in English. My native language does not use "r" as much. But what do I know about English? I learned it from my father, who taught it to himself by reading English novels in one of Joe's Gulags. [*Awkward pause while Alice struggles with how to respond.*] Have you discovered anything beautiful? [*Pointing into her notebook.*]

ALICE: Oh that? I've been thinking about mnemonics for passcodes.

[17] `$ grep '^.\{6\}$' /usr/share/dict/words | tr '[:upper:]' '[:lower:]' | sed 's/[abc]/2/g; s/[def]/3/g; s/[ghi]/4/g; s/[jkl]/5/g; s/[mno]/6/g; s/[pqrs]/7/g; s/[tuv]/8/g; s/[wxyz]/9/g' | sort | uniq | wc -l`

PAUL: [*Pointing to the drink:*] That poison will not help you. [*Produces a small pill bottle out of his shirt pocket, raises it to eye level, drops it, and then catches it with the same hand before it hits the bar.*]

ALICE: Haven't you heard? The *Ballmer Peak* is real! Or at least that's what I read on Stack Exchange.

PAUL: Pál Erdős. My brain is open.

PASTOR: Alice introduces herself and proceeds to explain all of her findings to Paul.

ALICE: ...and I just finished sorting the 14684 distinct codes by the number of words associated with them. That way, if I try the codes in order of decreasing word associations, then it will maximize my chances of cracking the code sooner than later.

PAUL: Yes, if codewords are chosen uniformly from all six-letter English words. Can I see the distribution of word frequency? [*Grabbing a napkin, stealing Alice's pen, and scribbling some notes.*] Using your method, after fewer than 250 attempts, there is a 5% probability that you will have cracked the code. After about 5700 attempts, there will be a 50% probability of success.

ALICE: [*Typing on her computer.*] That's only about 16 days!

PASTOR: An adversary with intermittent access to the lock—for example, after hours—could quite conceivably crack the code in less than a month.

PAUL: If there exists a method that allows the code-breaker to detect whether each successive two-digit subcode is correct before entering the next two-digit subcode,...

PASTOR: ...otherwise known as a "vulnerability"...

PAUL: ...[*annoyed about having been interrupted, even if by the disembodied voice of a narrator*] then the expected value for the length of time required to crack the code is on the order of minutes. [*Mumbling toward the fourth wall:*] That Pastor is more annoying than the SF.

ALICE: What?

PAUL: SF means "Supreme Fascist." This would show that God is bad. I do not claim that this is correct, or that God exists. It is just a sort of half-joke. There is an anecdote I once heard. Suppose Israel Gelfand and his advisor, Andrei Kolmogorov, were to both arrive in a country with a lot of mountains. Kolmogorov would immediately try and climb the highest mountain. Gelfand would immediately start building roads. What would you do?

ALICE: I would learn to fly an airplane so I could discover new mountain ranges. What about you?

PAUL: Some might say that *is* what I do. My friends might add that they pay for the fuel. But really, I just try to keep the SF's score low. How can we create mnemonics that are not vulnerable to your attack?

ALICE: Well, I guess the first thing to do is create a keypad layout that uses zero and one.

PAUL: Yes, but my academic sibling Pólya would say that we first need to understand the *problem*. Ideally, we want a keypad layout that produces an injective mapping from the six-letter English words into the natural numbers from zero to one million.

ALICE: Injective?

PAUL: Such that no two words produce the same code number.

ALICE: Is that even possible?

PAUL: I do not know. I believe this is an instance of the *multiple subset sum* problem, related to the knapsack problem.

ALICE: Ah yeah, I remember that from my algorithms class. It's NP-Complete, right?

PAUL: Yes, and likely intractable for problems even as small as this one. The total number of possible keypad mappings is 100 million billion billion. But it is easy for us to check the pigeons.

ALICE: Huh?

PAUL: The *pigeonhole principle*. For any subset of m letters within a word, there can be at most 10^{6-m} words that have that pattern of

letters. If there are more, then there must be a collision, no matter the mapping we choose.

Alice: Ah, I see. That's easy enough to check! [*Typing.*]

```
for m in range(2,6):
    hits = {}
    for word in words:
        for indexes in itertools.
    combinations(range(len(word)), m):
            key = tuple((word[i], i)
    for i in indexes)
            if key not in hits:
                hits[key] = 1
            else:
                hits[key] += 1
    max_hits = 10**(6-m)
    for key, h in hits.iteritems():
        if h <= max_hits:
            continue
        k = ['.' for i in range(6)]
        for c, i in key:
            k[i] = c
        print "".join(k), h - max_hits
```

So, there are fourteen five-letter suffixes like "inder", "aggle", and "ingle" that will all produce at least one collision. I guess there's no way to make a perfect mapping.

Paul: Gelfand advised Endre Szemerédi. This problem is reminiscent of Szemerédi's use of *expander graphs* in pseudo-random number generation. What we want to do is take a relatively small set of inputs (being the six-letter English words) and use an expander graph as an embedding into the natural numbers between one and a million, such that the resulting distribution mimics uniformity.

Alice: That sounds ... difficult.

Paul: Constructing expander graphs is extremely difficult. But I think Szemerédi would agree that interesting things rarely happen in fewer than five dimensions.

Alice: I am a pragmatist. How about we use a genetic algorithm to evolve a near optimal mapping?

Paul: Such a solution would not be from *The Book*, but it would provide you with a mapping.

Alice: What book?

Paul: The Book in which the SF keeps all of the most beautiful solutions.

Alice: Well, I think I'll try my hand at a scruffy genetic algorithm. I need a decent mapping if I ever want to publish this in PoC||GTFO!

Paul: What is PoC||GTFO?

Alice: It's... I guess it's a sort of bible.

Paul: Then the only difference between your Book and mine are the fascists who created them. Maybe we will continue tomorrow ... if I live.

Alice: [*Looking up from her keyboard.*] Can I buy you a drink? [*Paul has vanished.*]

Pastor: The moral of the story, dear neighbors, is *not* that these locks are inherently vulnerable; if used properly, they are in fact incredibly secure. We must remember that these locks are only as secure as the codes humans choose to assign to them. Using a phone keypad mapping on six-letter English dictionary words is the physical security equivalent of a website arbitrarily limiting passwords to eight characters.